Introduction to Statistics and Data Analysis: Basic plots in R

Dr. Niccole M. Pamphilis

Introduction

There are a lot of ways to create graphs in R (just like anything else in R). We are going to look at one of the more basic ways of creating plots to start with and we will add more details as we go on. Today we will look at the commands for creating bar charts, histograms, boxplots, and scatter plots. These are some of the common graphs you will use to examine, summarise, and show case your data.

For the examples that follow we will use the cleaned Scottish election data.

```
data4<-readRDS(file = "cleanedScot_data.rds")
```

Bar Charts

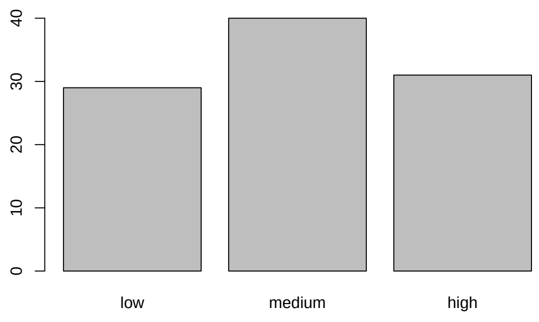
Recall that bar charts can be used to show the distribution of observations for nominal and ordinal level data. Here the height of the bars corresponds to the number or proportion of observations that fall in to each category.

In R before we can create a bar chart we need to create a table that records how many observations call in to each category for our variable. After we create the table this information is used by R to produce the bar chart.

```
##First, generate a table for the variable education.

table(data4$education)
```

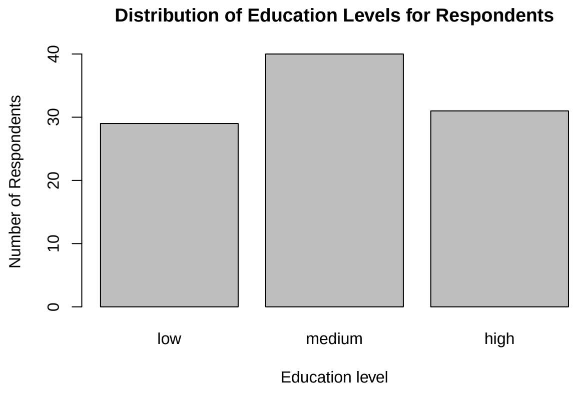
```
##
##    low medium   high
##     29     40     31
```

```
#Next, use the barplot command on the table.

  #Notice, that the barplot and table command can be run at the same time.

barplot(table(data4$education))
```

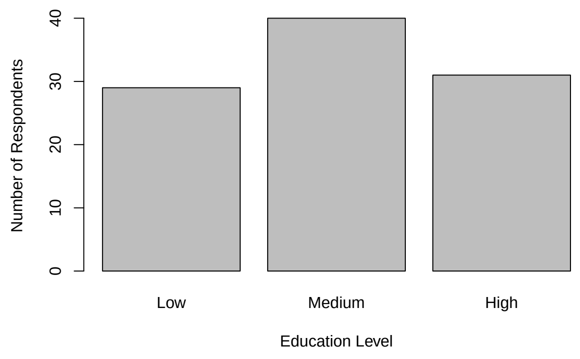


The bar chart above already has the labels for the categories applied because we added those data values when we cleaned our data set last week. But, we are missing an x-axis label, a y-axis label, and a title for our graph. We can add those to our graph too.

```r
#The xlab="", ylab="", and main="" options allow us to add titles

barplot(table(data4$education),
        xlab="Education level",
        ylab="Number of Respondents",
        main="Distribution of Education Levels for Respondents")
```



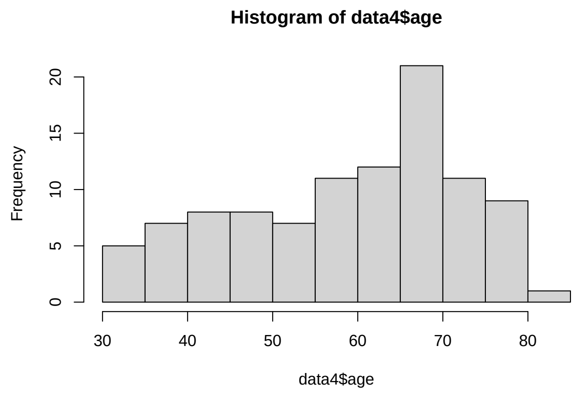
```r
#Our dataset already added labels to the data values. If we had not then R would show the
#number that corresponds to each category and we would need to add labels in the graph. To
#do this we us the, "name"" option as shown here.

barplot(table(data4$education),
        xlab="Education Level",
        ylab="Number of Respondents",
        names=c("Low", "Medium", "High"))
```



Histogram

To look at the distribution of observations for a histogram we can use the hist command in R.

```r
#The basic command is hist() with the selected variable

hist(data4$age)
```

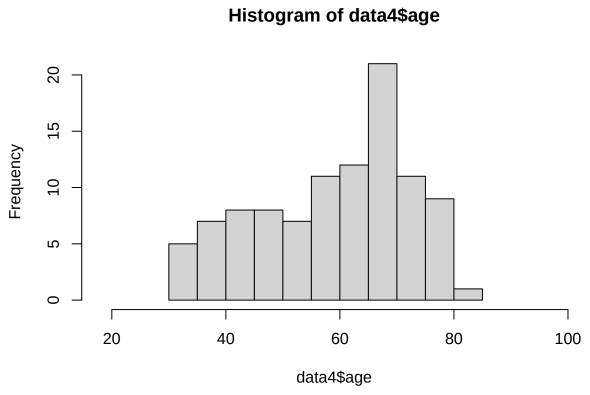


Extending the axis range

R, like most statistical programs will attempt to set the values shown on your axis range based on an algorithm of what will be best visually. However, sometimes we will want to increase or alter the range of our axis. We may want to do this to avoid having a truncated axis (omitted values) or because we want to make two separate graphs more comparable to each other and so we want the same range of values used.

Notice that in the graph above if we do not look at the values on the x-axis it appears that we have we observe a full range of possible values. Now, consider that respondents could have ranged from 18 through 100. We can adjust the x-axis to indicate that that values could have been as low as 18 using the xlim option.

```r
hist(data4$age,
     xlim=c(18,100))
```



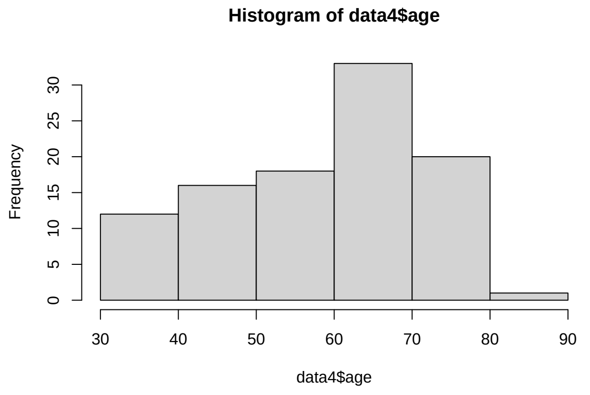
In the graph above we can now see that there are no observations at the low or high end of the possible values. This has implications for what our results will allow us to say later on in terms of external validity.

To adjust the y-axis in terms of size we can use the "ylim=c(,)" options. As with above we need to indicate the lower bound and the upper bound of our new set value range.

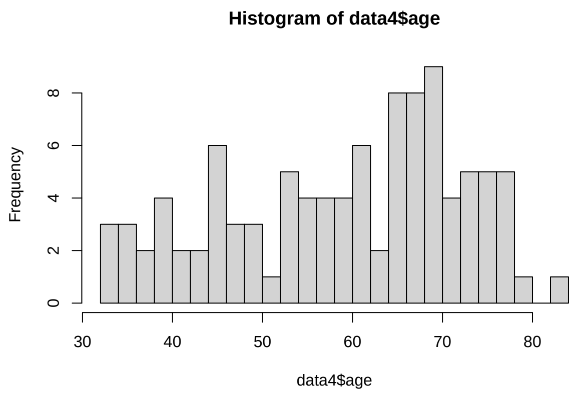
Breaks in Histograms

You can also adjust the number of bars or groups R breaks your data into using the breaks command. Keep in mind, R may not produce exactly the number of breaks you request if it cannot fist your data range into that set, but will get close.

```
#Decrease the number of bins

hist(data4$age,
     breaks=6)
```

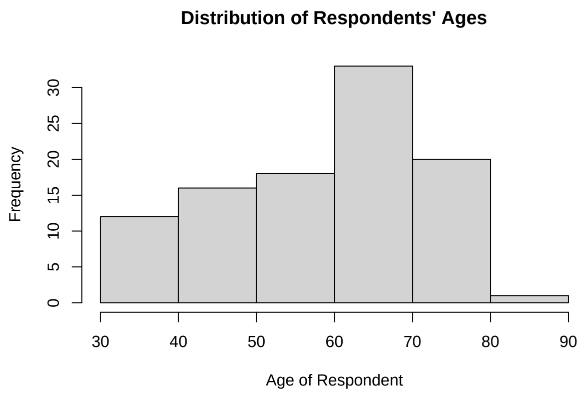


```
#Increase the number of bins

hist(data4$age,
     breaks=20)
```



Do not forget to clean up your graphs with adding labels.

```
hist(data4$age,
     breaks=6,
     xlab="Age of Respondent",
     main="Distribution of Respondents' Ages")
```



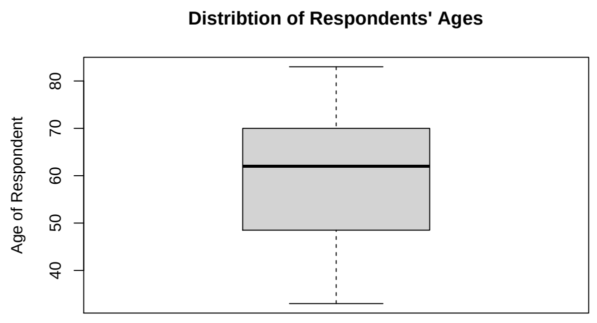
Boxplots

Another way to examine the distribution of your data is through the use of boxplots. Boxplots can be used to look at continuous level data (interval/ratio variables) as well as ordinal level data.

The middle line in the box represents the median value. The edges of the box indicate Q1 and Q3. The whiskers that come off the side relate to the range of common values observed in the data, while any points marked outside the whiskers correspond to outliers (values that do not fit the general pattern of the data).

The command for a boxplot is "boxplot", like our other commands from earlier you then select your variable of interest. And like earlier, you can add options for axis labels and a main title.

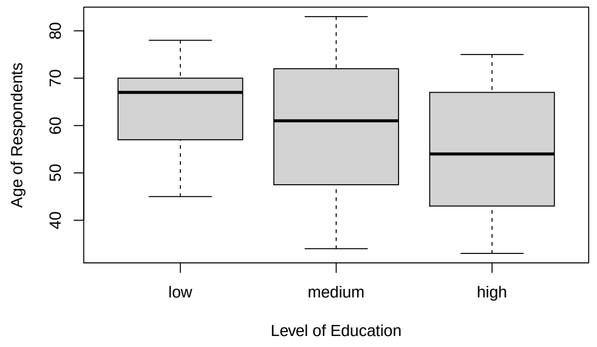
```
boxplot(data4$age,
        ylab="Age of Respondent",
        main="Distribtion of Respondents' Ages")
```



You can also create dual boxplots, where you look at the distribution of a variable across different groups. To do this you add one additional comment to your code a ~ followed by the grouping variable.

Below we have boxplots of age by education levels.

```
boxplot(data4$age ~ data4$education,
        ylab="Age of Respondents",
        xlab="Level of Education")
```

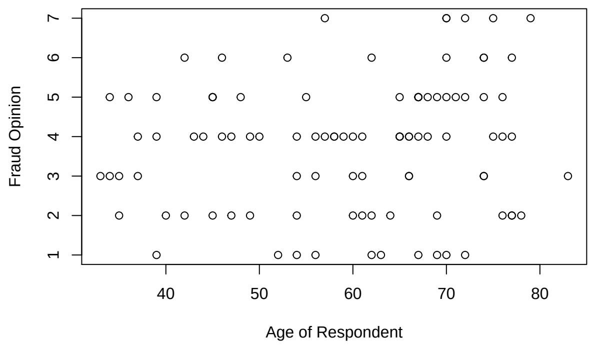


Scatter plots

As we progress in our research we are typically interested in finding patterns and relationships between variables. One of our first indications about patterns or relationships can be found through visual diagnostics. A common approach to looking at the relationship between to continuous level variables is to use a scatter plot.

To create a scatter plot we use the plot command. The first variable listed after the plot command will be plotted on the x-axis and the second variable will be plotted on the y-axis. As before, we can add labeling options too.

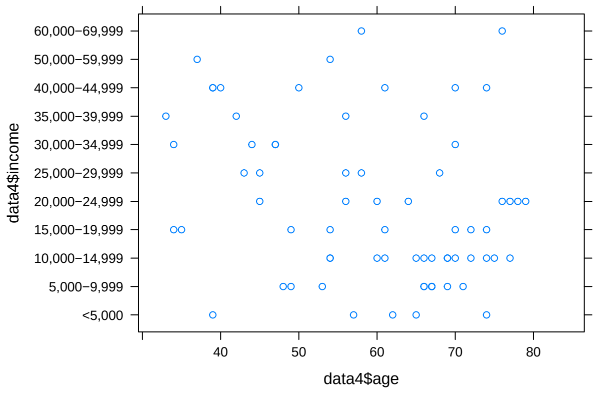
```
plot(data4$age, data4$fraud,
     xlab="Age of Respondent",
     ylab="Fraud Opinion")
```



Introducing Lattice

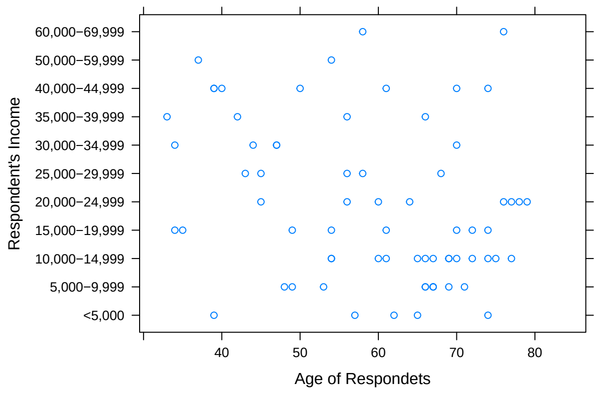
A basic scatter plot using lattice use the xyplot command. After that you list your y-axis variable followed by your x-axis variable, separated by a ~

```
library(lattice)

xyplot(data4$income~data4$age)
```



Like with our basic graph commands from last week we can add in titles for our x and y-axis.

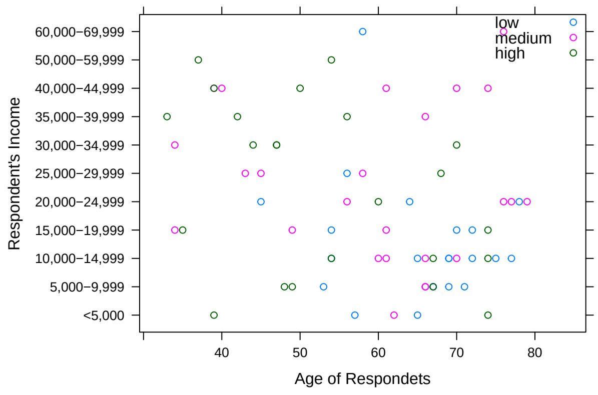
```
xyplot(data4$income~data4$age,
       xlab="Age of Respondets",
       ylab="Respondent's Income")
```



We can now also add a third variable to group by using the groups option. This allows us to look at patterns across a third variable.

You will also want to include the legend with the auto.key command.

```
xyplot(data4$income~data4$age,
       groups=data4$education,
       auto.key=list(corner=c(1,1)),
       xlab="Age of Respondets",
       ylab="Respondent's Income")
```

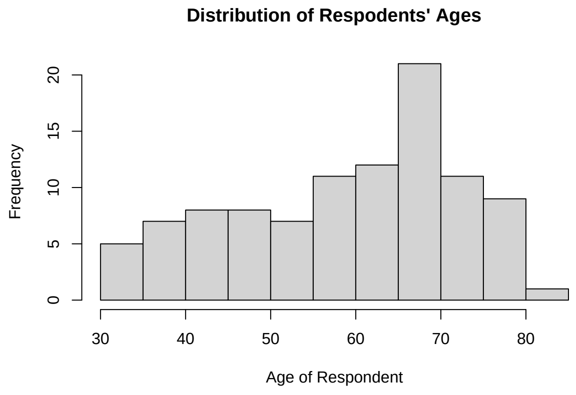


Histograms in Lattice

Using lattice we can also create slightly nicer histograms with added features compared to before.

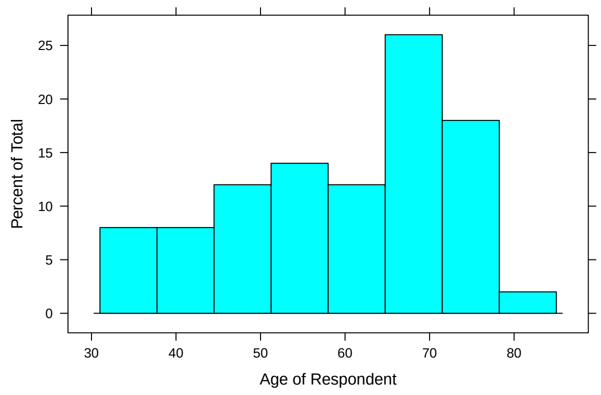
Let's start by looking at our histogram of age from last week using the basic command.

```
hist(data4$age,
     xlab="Age of Respondent",
     main="Distribution of Respodents' Ages")
```



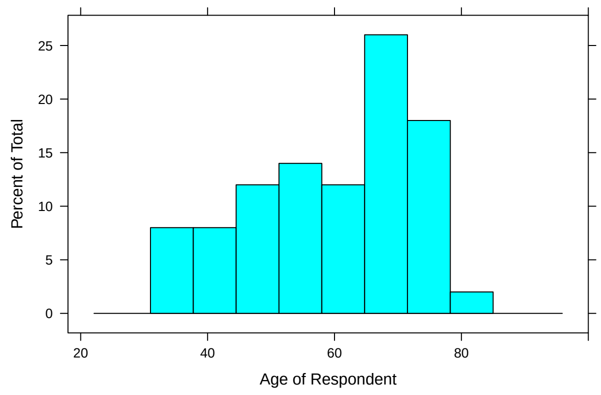
Now let's look at the same graph using lattice

```r
histogram(~data4$age,
          xlab="Age of Respondent")
```



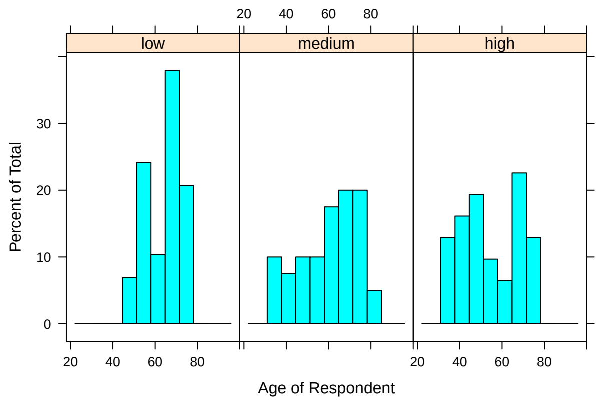
Now, with the histogram we get a slightly cleaner presentation, and as with before we can also now adjust the ranges of the x-axis if we desire.

```r
histogram(~data4$age,
          xlab="Age of Respondent",
          xlim=c(18,100))
```



We can also look at the distribution of the variable across a group variable, such as education using the lattice package.

```r
histogram(~data4$age| data4$education,
          xlab="Age of Respondent",
          xlim=c(18,100))
```

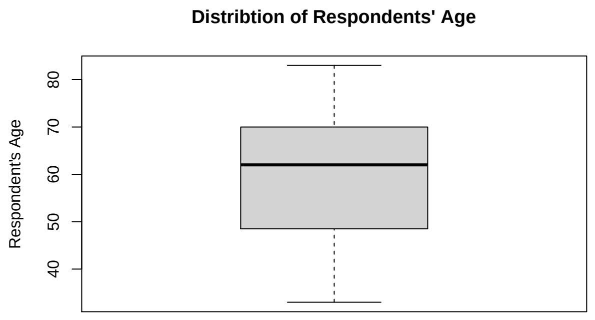


Boxplots in Lattice

Again, with boxplots we can improve our boxplots

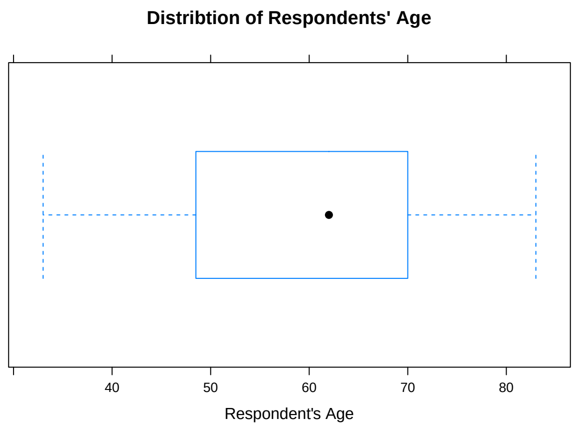
We can look at the boxplot of age, first using the basic command then using the lattice command.

```r
boxplot(data4$age,
        ylab="Respondent's Age",
        main="Distribtion of Respondents' Age")
```



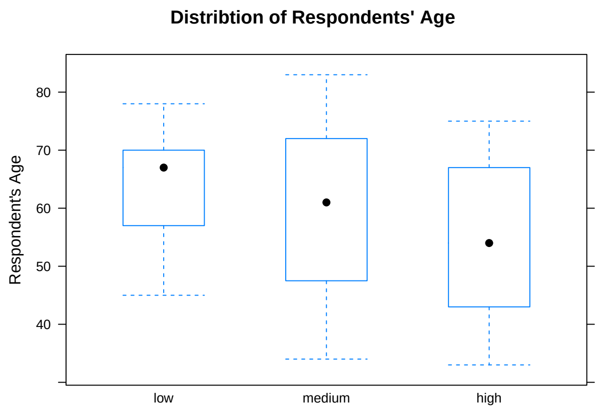
Now using lattice, which uses the bwplot (box and whisker plot) command:

```r
bwplot(data4$age,
       xlab="Respondent's Age",
       main="Distribtion of Respondents' Age")
```



You can also group your boxplots.

```
bwplot(data4$age~data4$education,
       ylab="Respondent's Age",
       main="Distribtion of Respondents' Age")
```

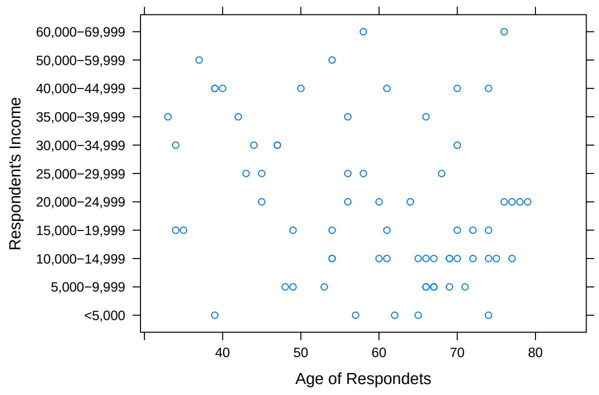


Changing your plotting sybmol

For increased clarity, instead of changing color options, you may choose to vary the plotting symbol in your graph. This can be done using the pch option. For a list of all options type help(pch) and scroll to the bottom.

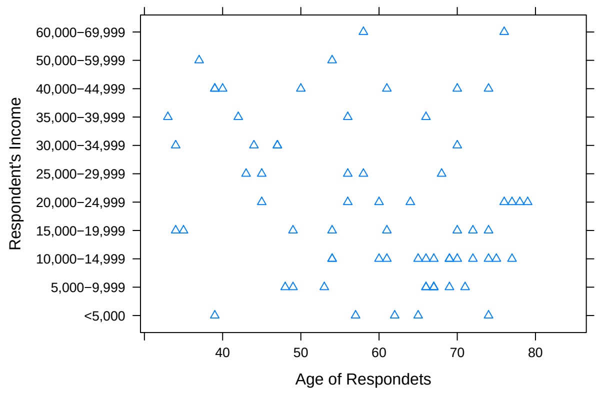
For a basic plotting character change we can look at our scatter plot from earlier.

```
xyplot(data4$income~data4$age,
       xlab="Age of Respondets",
       ylab="Respondent's Income")
```



Now, let's change the plotting character.

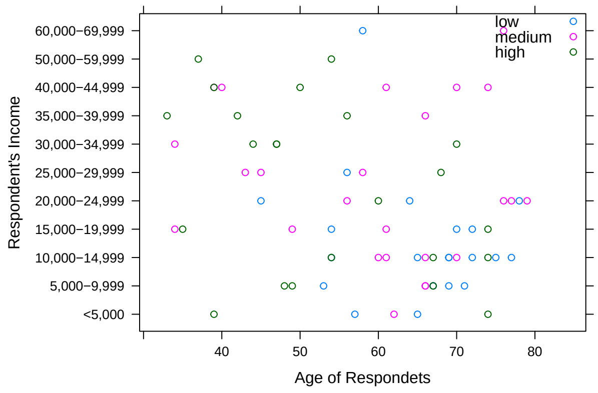
```r
xyplot(data4$income~data4$age,
       pch=2,
       xlab="Age of Respondets",
       ylab="Respondent's Income")
```



If we have multiple groups we could assign each a unique plotting character instead of color.

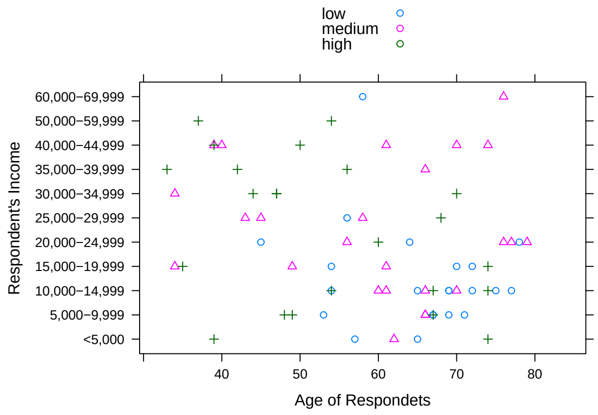
Let's look at the scatter plot from earlier again.

```r
xyplot(data4$income~data4$age,
       groups=data4$education,
       auto.key=list(corner=c(1,1)),
       xlab="Age of Respondets",
       ylab="Respondent's Income")
```



Now let's change the characters.To see the list of options for "pch" search in the help tab on the right (you will need to scroll down to find the images of the plotting characters with the numbers).

```
xyplot(data4$income~data4$age,
       groups=data4$education,
       pch=c(1,2,3),
       auto.key=T,
       xlab="Age of Respondets",
       ylab="Respondent's Income")
```

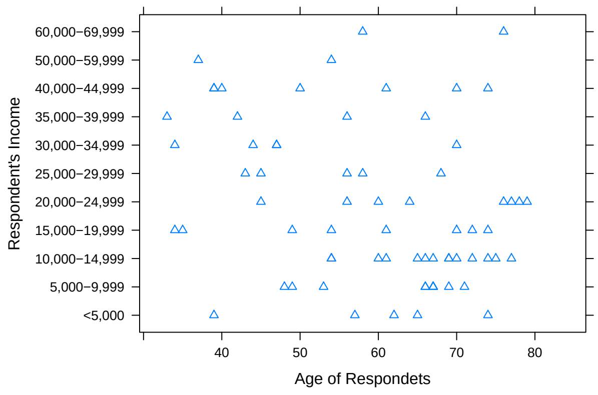


Aspect Ratio

One final point to consider is setting the aspect ratio of your graph. At times, it makes sense to ensure that units are equally space apart on the x-axis relative to the y-axis. To do this, we can set the aspect ratio when we create our graph.

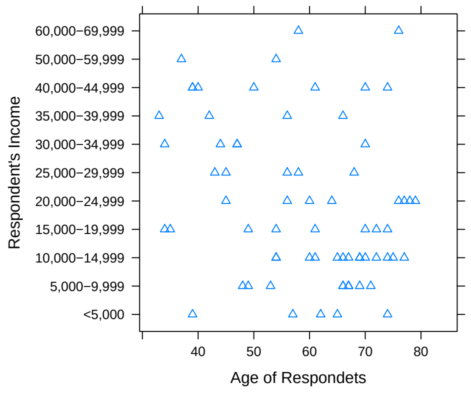
First a graph without the aspect ratio set

```
xyplot(data4$income~data4$age,
       pch=2,
       xlab="Age of Respondets",
       ylab="Respondent's Income")
```



Now, let's set the aspect ratio to 1.

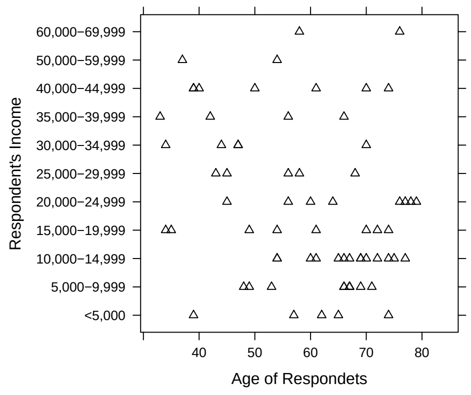
```r
xyplot(data4$income~data4$age,
       aspect=1,
       pch=2,
       xlab="Age of Respondets",
       ylab="Respondent's Income")
```



Recall that I had mentioned we generally want to avoid the use of color as it typically does not translate when printing/publishing. We can change the default colors used in our graphs with the "col" option.

Color options are referenced by name. If you want to use a common color name, it will likely be found in R. But there are many options, with less familiar names. To see a list of options you can Google "color options in R".

```r
xyplot(data4$income~data4$age,
       aspect=1,
       pch=2,
       col="black",
       xlab="Age of Respondets",
       ylab="Respondent's Income")
```



Respondent's Income
60,000−69,999
50,000−59,999
40,000−44,999
35,000−39,999
30,000−34,999
25,000−29,999
20,000−24,999
15,000−19,999
10,000−14,999
5,000−9,999
<5,000
40
50
60
70
80
Age of Respondets